Closed Book Examination

Appendices attached

One and a half hours

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

Model-Based Software Design

Tuesday 27th January 2009
14:00 – 15:30

Please answer any TWO questions from the FOUR questions provided

The use of electronic calculators is NOT permitted.
1. a) In terms of the definition of models for use in software development, describe the meaning or role of the following terms and how they relate to each other: abstract syntax, graphical and textual concrete syntaxes, invariant, Ecore, OCL, Emfatic, Ecoretools. (10 marks)

b) Consider the Emfatic model of a bank shown in Appendix A and answer the following questions:

i) What is the meaning of the multiplicity used on line 44? (1 mark)

ii) What is the meaning of the multiplicity used on line 63? (1 mark)

iii) What is the effect of the invariant on line 70? (1 mark)

iv) What is the effect of the invariant on line 76? (1 mark)

v) From an implementation viewpoint, what is the effect of the opposite clause in lines 41 and 96? [Hint: opposite clause starts with a #] (1 mark)

c) Applications are to be developed that target a multi-screen Java Swing interface. To allow these applications to be generated within a model-based development process, outline an Emfatic meta-model of the target Java Swing environment. (5 marks)

2. a) Critically evaluate the aspects of a modelling notation that need to be considered when evaluating alternative notations for use in the development of software applications. (5 marks)

b) Write an Emfatic model of the following scenario, your model should be written so that it is suitable for input into the EMF implementation generation tools. (10 marks):

A library contains books that must have both a title and one, or more, authors. As an author may have written more than one book, the library allows searching of its catalogue by both book title and author. The library gives all books a seven character identifier and classifies books into single works and part works; single works only have the properties of books, whereas part works have additional properties of a numeric volume year and number.

c) The above model covers the domain aspects of an application. If an application is to be fully described by models and an implementation generated from these, what other aspects of an application will need to be modelled? For each aspect that you mention, you should give an indication of the notation that could be used to model it and how concise a model would be. (5 marks)
3. a) State the three types of transformation that exist in model-driven software
development, and describe the role of each. (6 marks)

b) By considering the Emfatic model of a bank in Appendix A and the Emfatic
model of part of a website in Appendix B, outline an Atlas Transformation
Language (ATL) transformation that will generate a website interface that
supports the display and maintenance of the information for a bank. (10 marks)

c) Critically evaluate the techniques that could be used to transform the model
that results from your transformation in the previous part, into implementation
code, configuration codes, html pages, etc. (4 marks)

4. a) What are the four modelling levels that exist in model-driven software
development? (4 marks)

b) How do each of the levels that you identified in part a) relate to each other?
(1 mark)

c) Describe the purpose of two of the modelling levels that you identified in part
a). (4 marks)

d) An application is to be developed that parses a domain specific language
(DSL), in terms of a model-based development environment, discuss the
options that are available for the development of tooling to process this
language. (6 marks)

e) Critically evaluate why, in the model-driven development domain, basing the
implementation of tooling on meta-models produces generic tools that can be
used across many projects. Your answer should give examples of this
approach being used and discuss why this approach is used. (5 marks)

END OF EXAMINATION

Appendices A and B follow on the next three pages
Appendix A – Emfatic Model of a Bank

1  package EBank;
2  package People {
3      enum MaritalStatus { 
4          Single; Married; Divorced; Widowed;
5      }
6
7      @inv(name=agePositive, spec="age > 0")
8      abstract class Person {
9          attr String[1] name;
10         attr int[1] age;
11         attr MaritalStatus[1] maritalStatus;
12      }
13
14      class Female extends Person {
15          ref Male[0..1]#wife husband;
16      }
17
18      class Male extends Person {
19          ref Female[0..1]#husband wife;
20      }
21
22  }
23
24  package EBank {
25
26      @inv(name=phoneNumberLength, spec="name.size() = 10")
27      class Bank {
28          attr String[1] name;
29          attr String[1] phoneNumber;
30          val Branch[+]#partOf branches;
31          // derive staff;
32      }
33
34      @inv(name=uniqueSortCode,
35            spec="partOf.branches->select(
36                b | b.sortCode = self.sortCode)->size() = 1")
37      class Branch {
38          attr String[1] sortCode;
39          ref Customer[*]#branch customers;
40          val Account[*]#heldAt accounts;
41          // derive branch staff
42          ref Bank[1]#branches partOf;
43      }
44
45      enum AccountCurrency {
46          GBP; EUR; USD;
47      }
48
49      @inv(name=accountNumberPositive, spec="accountCode > 0")
50      @inv(name=accountNumberUnique,
51            spec="heldAt.accounts->select(
52                a | a.accountNumber = self.accountNumber)->size() = 1")
53      @inv(name=interestRatePositive, spec="interestRate >= 1")
@inv(name=x,
   spec="not overdraftLimit.oclIsUndefined()
   implies overdraftLimit >= 0")
abstract class Account {
   attr String[1] accountNumber;
   attr Date[1] openingDate;
   attr double[1] interestRate;
   attr double[0..1] overdraftLimit;
   attr AccountCurrency[1] accountCurrency;
   val Transaction[*] transactions;
   ref Branch[1]#accounts heldAt;
   ref Customer[1]#accounts ownedBy;
}

@inv(name=y,
   spec="not overdraftLimit.oclIsUndefined()")
@inv(name=interestFixed, spec="interestRate = 2.5")
class CurrentAccount extends Account {
}

@inv(name=z,
   spec="overdraftLimit.oclIsUndefined()")
@inv(name=interestVariable,
   spec="(1.0 <= interestRate) and (interestRate <= 14.5)")
class SavingsAccount extends Account {
}

enum TransactionType {
   Withdraw; Deposit;
}

@inv(name=amountPositive, spec="amount > 0")
class Transaction {
   val TransactionType[1] transactionType;
   attr Date[1] date;
   attr double[1] amount;
}

class Customer extends People.Person {
   ref Account[*]#ownedBy accounts;
   ref Branch[1]#customers branch;
}
Appendix B – Emfatic model of a web interface

```java
package Web;

class Website {
    val Page[+] pages;
}

class Page {
    val PageElement[+] elements;
}

abstract class PageElement {
}

class Table extends PageElement {
    val TableRow[+] rows;
}

class TableRow {
    attr String[1] label;
    attr String[0..1] value;
}

class Form extends PageElement {
    val FormField[+] fields;
}

class FormField {
    attr String[1] name;
    attr String[0..1] currentValue;
}
```